**Bach Le**

**ANALYSIS OF DIFFERENT MATRIX MULTIPLICATION ALGORITHMS WITH PARALLEL COMPUTING BASED ON PERFORMANCE AND SCALABILITY**

Speed up and efficiency versus the increasing level of parallelism

with regard to 4 different algorithms at different sample sizes (N).

**1. In regard to different algorithms**

Based upon the above graphs, we can quickly identify that the IKJ algorithm is the most superior algorithm at both performance and scalability. It yields the most speed up and efficiency at every level of parallelism and at every sample size, except for N=1000 with 48 processors. Following the IKJ, the IJK is the second best algorithm due to its stability at small matrix size while still maintain its performance at larger matrix size at every level of parallelism. At a larger matrix size, the mxm2 algorithm matches the performance of IJK algorithm. At the last place is the JKI algorithm with the worst performance at high level of parallelism at almost every matrix size, except N=1000.

However, it is interesting to note that while working with a large matrix size, the performances of different algorithms are not too distinctive, but with a small matrix size, the mxm2 and JKI algorithms have a detrimental effects on the performance at high level of parallelism. To be specific, at N = 100, working with more than 16 processors will have little to no effect on the performance when comparing to the sequential run time. The same is applied for the mxm2 algorithm at N=1000.

**2. In regard to matrix size**

Based on the above results, the size of matrix play an important role in the diversity of performances of different algorithms. At a small matrix size, the choice of algorithm is crucial to the run time of the multiplication – IKJ and IJK are much better than mxm2 and JKI. However, at N = 1000 and 3000, the difference of efficiencies is trivial, except for the mxm2 algorithm at N = 1000. Up until N = 5000, we can observe the diversity again with IKJ algorithm offers the best speed up and efficiencies through out all level of parallelism.

More than that, increasing matrix size also increases the efficiencies of all algorithms. To be specific, at N = 100, we observe significant drop in efficiencies in high parallelism. All algorithms drop to almost 0 efficiency while working with 48 processors with N = 100. However, at N = 1000, 3000, 5000, the efficiencies of all algorithms is relatively stable, with IKJ algorithm stays above 0.8 efficiency at all scenarios except N = 1000 with 48 processors. Thus, we can conclude that for each matrix size, there is a level of parallelism that maximize performance, yet if we exceed this level, the performance will take a hit.